Please attribute to : Paul Sutton : http://www.zleap.net : @zleap14 : zleap@zleap.net

HOW_TO_1 **CREATE A WINDOW IN PYTHON / TKINTER**

Once you are familiar with creating programs in python that run from the console the next step up from this is to creatre programs that will run in a window.

The following program produces a Window on the screen and sets various attributes

```
#!/usr/bin/env python
import Tkinter # note use of caps
from Tkinter import *

window = Tk()
window.title('GUI Tkinter 1')
window.geometry("300x250") # w x h
window.resizable(0,0)

window.mainloop()
```

Listing1

fig: 1

If we break this down we get :

#!/usr/bin/env python	This tells the interpreter to use python
import Tkinter **from** Tkinter **import** *	This imports the Tkinter module we need
window = Tk() -	Create window object
window.title('GUI Tkinter 1')	Define the Window title
window.geometry("300x250") *# w x h*	Set window size width x height
window.resizable(0,0)	Define if we can resize the window or not
window.mainloop()	This displays the window

HOW_TO_2 **ADD A LABEL**

Carryong on from the first How to, we are now going to add a lable to the window we created earlier.

To do this we need to load the code we produced before up and add an extra two lines of code. The new code should be added BEFORE the window.mainloop()

```
#define labeles
box1 = Label(window, text="Entry 1: ")

#place labels
box1.grid(row = 1 , column = 1, padx = 5, pady = 5)
```

For this we are using the Grid method to place the label on the screen.

About the code :

1. Defines the label and what text it contains
2. Places the label within the window object.

Getting the position right is harder. Once you start adding more object it gets a little easier.

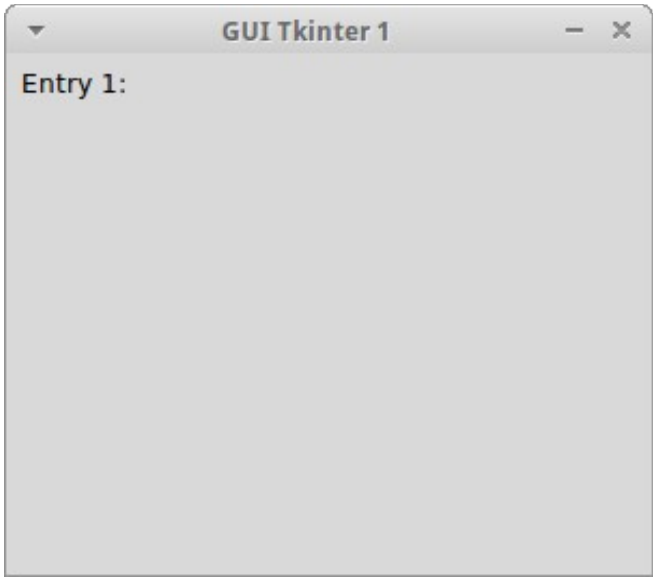


fig 2

HOW_TO_3 **ADD A BUTTON**

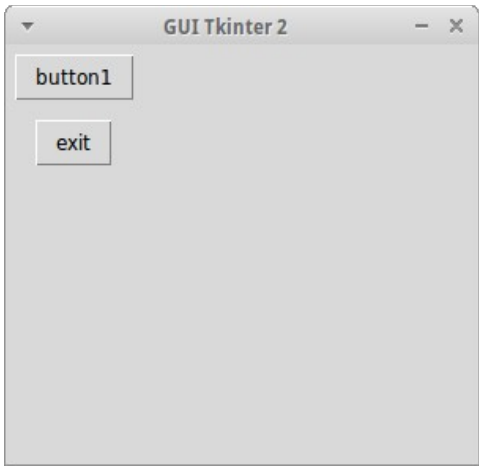
Now that we are able to add objects to the wi ndow. We can now add more interactive components.

1. Delete the code we added to place the label on the window
2. Put this code above the window.mainloop()3

```
def btn1():
        print ("button pressed")

btn_tog2 = Button( window, text ='button1',
command=btn1)
btn_exit = Button( window, text ='exit',command=exit)


btn_tog2.grid(row = 1, column = 1, padx = 5, pady = 5)
btn_exit.grid(row = 2, column = 1, padx = 5, pady = 5)
```



Here we are introducing functions, you MUST define a function BEFORE you call it from else where in theprogram.

HOW_TO_4 COMBINE LESSONS 2 AND 3

Now that we are able to add a lable and a button to a window, we can start to make applications more user friendly. While buttons have labels the purpose of this lesson will simply be about adding both. So the buton has text with a description of what the button does.

```
#define functions for button(s)
def btn1():
        print ("button pressed")


#create button object
btn_tog2 = Button( window, text ='button1', command=btn1)
btn_exit = Button( window, text ='exit',command=exit)

#place button object
btn_tog2.grid(row = 1, column = 2, padx = 5, pady = 5)
btn_exit.grid(row = 2, column = 2, padx = 5, pady = 5)

#define labeles
button1 = Label(window, text="press button")
button2 = Label(window, text="exit program")

#place labels
button1.grid(row = 1, column = 1, padx = 5, pady = 5)
button2.grid(row = 2, column = 1, padx = 5, pady = 5)
```

Note a few things here, I have placed the objects explicitally on the window.

Column 1 has labels , Column 2 has buttons.

I have also tried to name the objects logically so programs are easier to debug later

not use of # lines for comments, It is good practicer to add comments to code.

There is no need to place a table here to explain what the code does, as we have simply combined what we have learnt before. 3

HOW_TO_5 SUMMARY SO FAR

So far we have learnt to:

Create a window.geometry
Add objects to a window, namely :

a label
a button

When a button is pressed do something.

So far we have learnt to:

Add a label
Add a button

When a button is pressed do something.

HOW_TO_6 MAGIC 8 part 1

Now that we can add objects we can add other things too.

What we will do first however is create a program that takes text input and then generates a random response, e.g question and answer program. This program was created by Tom Brough.

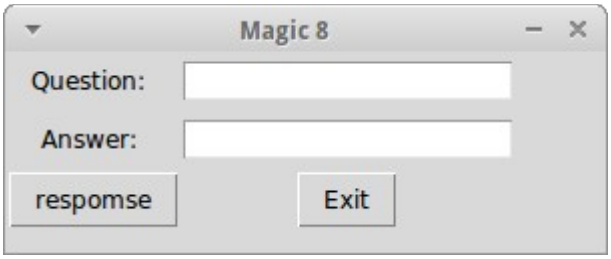
HOW_TO_7 MAGIC 8 part 2

The above program takes input in the form of raw input and then generates a random response.

Rather than create a lesson that simply teaches you how to create a text input box this lesson will do this but also build on what we have already covered.

Staying with the above program, we need to do several things

1. Create the elements we need to create a window on the screen
2. Crate the objects we need
    - Text boxes (one for input text and one for output text)
    - lable
    - Buttons
3. Place these on the screen
4. Tidy things up a little



Lets look at the program in more detail

```
import random
import Tkinter # note use of caps
from Tkinter import *
```

The main difference here is the addition of the ramdom module otherwise we are importing Tkinter as before

```
#set up
window = Tk()
window.title('Magic 8')
window.geometry("300x100") #wxh
window.resizable(0,0)
```

As before we define the window, you will notice I have made the window size 300 x 100. This makes the window a nice size around the program buttons and text entry / output boxes

We now define the responses we want and store these in an array called RESPONSES

```
RESPONSES = ["It is certain",
             "It is decidedly so",
             "Without a doubt",
             "Yes definitely",
             "You may rely on it",
             "As I see it yes",
             "Most likely",
             "Outlook good",
             "Yes",
             "Signs point to yes",
             "Reply hazy try again",
             "Ask again later",
             "Better not tell you now",
             "Cannot predict now",
             "Concentrate and ask again",
             "Don't count on it",
             "My reply is no",
             "My sources say no",
             "Outlook not so good",
             "Very doubtful"]
```

As Discussed earlier functions get defined first so we now define a function that calls up a random response from the array.

```
def response():
        x = random.choice(RESPONSES)
        #print x
        circletext2.delete(0, END) # clear prev output
        circletext2.insert(0,str(x))
```

Note a few things here

x is a variable that now stores the responses
note I have commented out with a # print x, (this was used to test and left in so it can be used later) uncomment while testing your programs

The next line simply clears the out put text box (keeps program tidy) you don't need it but it keeps things in good order
The next line does all the work, and inserts the response variable x as a string (str)

Once this is done we can start with the actual gui design.

```
#define labels - cannot share same name as function
box1 = Label(window, text="Question: ")
box2 = Label(window, text="Answer: ")

#place labels
box1.grid(row = 1, column = 1, padx = 5, pady = 5)
box2.grid(row = 2, column = 1, padx = 5, pady = 5)

#define entry box

circleVar = StringVar()
circletext = Entry(window, textvariable=circleVar)

#define out box

circleVar2 = StringVar()
circletext2 = Entry(window, textvariable=circleVar2)


#display boxes
circletext.grid(row = 1, column = 2,)
circletext2.grid(row = 2, column = 2,)

#define buttons

response = Button( window, text ='respomse', command=response)
exitbtn = Button( window, text ='Exit', command=exit)

#place buttons

response.grid(row = 4, column = 1, padx = 1, pady = 1)
exitbtn.grid(row = 4, column = 2, padx = 1, pady = 1)

#display window

window.mainloop()
```

IMPROVING MAGIC 8

Now we have the basic program working, we can look in to making a few enhancements. The main flaw with the above is

- If you don't type anything you get a respose
- If you simply type a string of numbers you get a response.grid

Which clearly isn't very helpful.

The following program, tests an input contains letters.

```
letter = raw_input("Enter your name ")
i = letter.isalpha()
while i != True:
        print("Input MUST use letters")
        letter = raw_input("Name ")
        i = letter.isalpha()
print letter
```

This can't be JUST integratred as it, lets look at the code in our magic 8 program we need to integrate this with.

```
def response():
        x = random.choice(RESPONSES)
        #print x
        circletext2.delete(0, END) # clear prev output
        circletext2.insert(0,str(x))
```

To start off have added the code to check if what you have put in is actually text. However using the function isalpha() doesn't work here as it seems to detect the spaces in your question and throws up an error.

So to get round this I did something slightly different

```
def response():

    msg = "error : must be a text value"

    i = circletext.get()
    y = i.isdigit()
    if y == True:
        circletext.insert(0,(msg))

    else:
        x = random.choice(RESPONSES)
        circletext2.delete(0, END) # clear prev output
        circletext2.insert(0,str(x)) # insert response
```

What I have done here is detect if the text in the text box is numeric, if it is then clearly it is not a question as such and it then gives an error.

This still leaves the issue of if the text input box has nothing in it then it will still give a response.

To do this

```
def response():

    msg = "error : must be a text value"

    i = circletext.get()
    y = i.isdigit()
    l = len(circletext.get())
    print l
    if y == True or l == 0:
        circletext.insert(0,(msg))
```

So bascially what we are saying here is that if y is a number or y is equal to zero length then insert the string stored in msg to the text box.

To clear up further we can add a new button to clear both the input and output boxes

We can start by creating a new function for this

```
def clear():
        circletext.delete(0, END) # clear input box
        circletext2.delete(0, END) # clear output box
```

then add a new button

```
clear = Button( window, text ='Clear', command=clear)
```

and place it on the canvas

```
clear.grid(row = 4, column = 2, padx = 1, pady = 1)
```

As I have put this in column 2 then I have moved the exit button to column 3

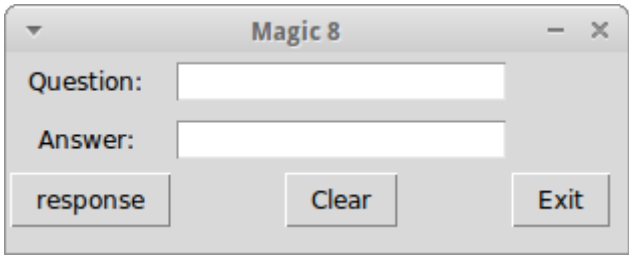
```
exitbtn.grid(row = 4, column = 3, padx = 1, pady = 1)
```

Other than that, we should have 3 buttons.

However if you run it then the buttons seem to be right on the edge so I have changed

```
window.geometry("300x100") #wxh
window.geometry("310x100") #wxh
```

to add that little but more width



I have also corrected a typo, so it now says response.